The Study of Rare Event Systems' Behavior Using Simulation

Sudeep Shrestha[1], Kuodi Jian[2]

[1]Metrostate University

[2]Information Computer Science, Metropolitan State University, Saint Paul, USA

[1]sudeep494@gmail.com ; [2]kuodi.jian@metrostate.edu

Abstract

Simulation is a useful tool in the design process when building a large-scale system. It helps to analyze several design factors like feasibility, cost, impact, outcome and risk in advance. Simulation is resource intensive when systems become large and complex. Especially, when we try to simulate a special type of systems called "rare event systems", the determination of the probability of such rare events with reasonable confidence interval can be a challenge in terms of computational resource, time, and budget. In this type of systems, the chance of the interested event's occurrence is slim. But the consequence of that is severe. Examples of this type of systems include: space exploration endeavors, air traffic control systems, nuclear reactors control systems, etc. Any rare accidental events in such systems could mean a missed opportunity to find a life in space, the loss of human lives, or the demise of entire life in the planet. The challenge of simulating this type of systems stems from the fact that we dare not miss important cases (have to cover those rare but important possibilities even though some of these are very low, thus a lot of computing time). In this paper, we will present a general-purpose rare event simulation framework package called "SPLITSIM" that is developed and promoted (distributed as open-source software) by us to address rare event simulation challenges; we will give descriptions about the architecture of our software design and give comments on essential parts of our implementation source code; we will also show features and advantages of our open-source package.

Keywords

Algorithms; System Design; Optimization; Simulation; Rare Time Events; Simulation Packages

Introduction

When designing any large-scale system, one needs to consider several factors such as cost, impact, lifespan, risks etc. While engineers and designers go to great lengths in making sure that each factor gets incorporated in their design, it is really hard to predict the actual outcomes. This is where simulation comes in. By simulating the system, one can predict if design criteria are really met. The nature of the rare event systems makes it an important topic in the area of system design and in the field of simulation. The fruit of this topic offers real benefits in terms of resource saving and (in some cases) life savings.

In the last few years, a lot of work has been focused on coming up with different approaches to conduct rare event simulation [1, 2, 3, 4, 5]. After surveying many researches, we conclude that there are mainly two methods that efficiently simulate rare events: Threshold Sampling and Threshold Level Splitting [2, 6, 7, 8]. Compare to the threshold sampling, the threshold level splitting is a breakthrough in the field of rare-event simulation since it produces low variance and it does not introduce bias. However, there aren't many general-purpose simulation software packages available that use the splitting concept. We find that (in general) there seems to be a lack of readily available software that can be used for simulating rare-events. Some researchers were conducted by Villen-Altamirano and Villen-Altamirano[5], which led to the simulation software known as ASTRO. However very little documentation can be found on the project and also it does not clarify if it has a plug-in architecture. This makes integrating with such software very hard. Furthermore, no enhancement can be done to the project since it is not open source.

In this paper, we present a software package that implements the concept of splitting to simulate rare events. The software framework and the package are named "SPLITSIM" which stands for "threshold level SPLITting software for cost effective rare event SIMulation." SPLITSIM offers a platform for anybody wishing to simulate a rare event in systems. There are some unique features in our proposed software framework: 1) it uses a newer approach [1] of splitting using number of successes to speed up the simulation, thus it promotes new concepts in the field of rare

event simulation, 2) it is open source software that encourages the continued improvement of the software by a large user community, 3) it offers user friendly features such as real-time display, graphic outputs, and level of interest analysis which has never been seen in other rare event simulation software.

There are many uses of our SPLITSIM software: 1) customers who just need off-the-shelf simulation software to solve system design problems; 2) simulation software developers who are interested in the inner workings of the level splitting; and 3) simulation software companies trying to minimize the cost for developing the rare event simulation software since our SPLITSIM is built with popular language Java and offers flexibility and extendibility that can be used to integrate with other external software components.

In section 2, we introduce the basic concepts of rare events and the threshold splitting; present the details about the SPLITSIM software framework. In section 3, we illustrate the usage of our SPLITSIM framework by applying it to an example. In section 4, we summarize our work.

The SplitSim Software Framework

In this section, we introduce the framework of the SPLITSIM software (because of the space restriction, we present only part of our source code in this paper. If a reader is interested in the complete code, please contact the authors by using the email addresses supplied). The SPLITSIM framework and its implementation (SPLITSIM package) are an effective tool to study the behavior of rare event systems. SPLITSIM relies on Threshold Splitting algorithm based on fixed number of successes [1] to conduct the simulation to determine the rare-event probability. In the following, we will introduce the details about this tool.

SplitSim Overview

SPLITSIM is a simulation tool written with JAVA language and distributed as open-source software with an Open BSD License. Open BSD guidelines state that the code can be used or redistributed with or without modification, provided that certain conditions specified by the Open BSD organization website [7] are met. As mentioned before, SPLITSIM uses the concept of threshold splitting, which efficiently yields the rare event estimator.

The Basics of Rare Events and the Threshold Splitting

In this paper, rare events are defined as events that have very high risk level but very low probability of occurrence. Such events might have probability of occurrence as low as 10^{-100}. This means that to accurately determine their probability, one has to perform 10^{100} trial runs for each occurrence of the event. This translates to even more runs in order to calculate a rare event probability with a reasonable amount of confidence interval. Rare events can occur in almost all systems. However, their importance is only manifested in those systems that will cause enormous damage when something goes wrong. A good example of rare event would be an event in a nuclear reactor where the temperature of the core exceeds the threshold that results in a nuclear disaster.

Threshold splitting is a technique to speed up the simulation. In this technique, multiple levels of thresholds are set for the simulation. Each level serves as a milestone until the maximum threshold is reached. The maximum threshold is the threshold at which the rare event of interest occurs. Thus, each threshold represents a level within the range that the system supports. These levels are referred to as threshold levels. Each threshold level simulated will yield a result with desired degree of variance. Once the targeted degree of variance is met for a level, the next levels of simulations are conducted from the preceding threshold level. Using this technique, we can get estimations that have considerably low variance since the degree of variance is maintained at each level. This technique also makes it possible to simulate a rare event in considerably shorter time. Figure 1 from Chen and Shortle [6] shows the basic concept of level splitting.

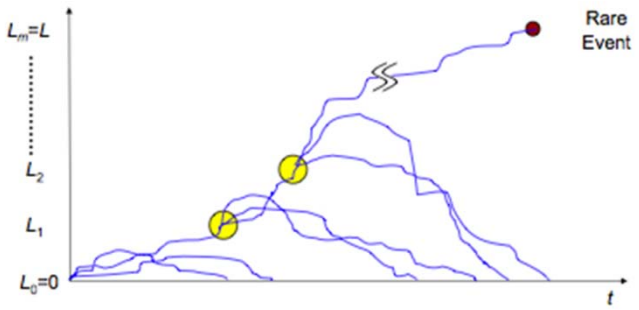

FIGURE 1 THE BASIC CONCEPT OF LEVEL SPLITTING

In Figure 1, the optimal threshold is represented as L_0, maximum threshold is represented as L_m and the new threshold levels are represented as L_1, L_2….. L_m. This technique works by simulating one level at a time

starting from the optimal threshold. First, we simulate the system to find out if the system reaches L_1 from L_0. Once enough information is gathered about this first level, we move on to the next level. The thing to consider here is that by now we already know the likelihood of the system reaching L_1 when started from L_0, so there is no need to do that again. Next, we simulate the system to find out if the system reaches L_2 when started from L_1. By doing so, we basically eliminate the time that we would spend simulating the rise from L_0 to L_1if this technique is not used. Repeat this process until we are in the very last level where we simulate the system to see if it rises to a threshold of L_m from L_{m-1}. This whole process is what we define as threshold level splitting. Figure 1 shows the path of the system as it reaches the rare event at Level L. The rare event threshold in this case is divided into m threshold levels. This means that the system state starts at level L_0 and at every level it conducts 3 sample runs until it hits level L_m where the rare event occurs.

The SPLITSIM (or SplitSim) Architecture and the Core Algorithm

The major components in the SplitSim framework can be categorized into three sections: UI Classes, Core Classes and Integration Classes as shown in Figure 2.

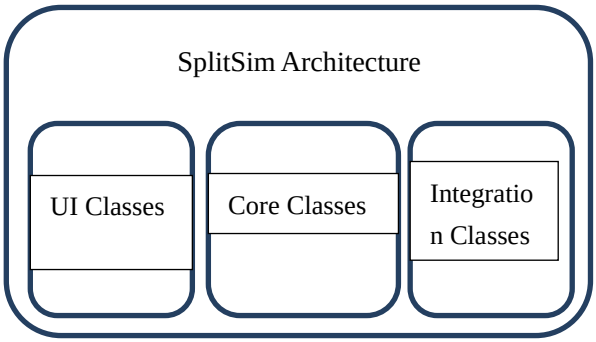

FIGURE 2 SPLITSIM ARCHITECTURE

The UI classes make it easier to collect simulation parameters and displays results and real-time information regarding the simulation. The core classes conduct the actual simulation. The integration classes make it possible for external systems to integrate with the SplitSim framework.

The core algorithm for fixed number of successes considerably simplifies splitting approach as opposed to other approaches where a better knowledge of the system is required. For instance, for fixed effort approach, the number of starting states must be chosen so that at least one run reaches the next threshold. The same applies for fixed splitting and RESTART because the number of splits must be so chosen that the threshold rise is never suppressed at any level due to lack of the sample size. SplitSim basically uses the count of the split provided to split the threshold range into that many sublevels. For each level, simulation is conducted so that the system reaches the next threshold exactly the same amount of times. This is based on the fixed number of success count that is provided to SplitSim. Failure, on other hand, is defined as runs that drop to the initial start state or to the truncation level provided to the system. This process is repeated until the rare-event occurs. The core algorithm used by the SplitSim framework is captured by the algorithm in List 1.

LIST 1 THE CORE ALGORITHM FOR THE SPLITSIM FRAMEWORK

```
==========================================

for (int level=1;level<=thresholdCount;level++){

SplitSim.startSimulation();

lastLimit= (SplitSim.thresholds.get(level-1)).getUpperLimit();

doubleupperLimit = lastLimit + thresholdJump;

doublelowerLimit = lastLimit - tBias*thresholdJump;

if ( lowerLimit<startThreshold){

lowerLimit=startThreshold;

   }

SplitSim.thresholds.add(newThreshold(level,upperLimit,lowerLimit))
;

while (SplitSim.thresholds.get(level).getHitCount() <hitMark){

        SplitSimSystems = newSplitSimSystem();
        s.initialize(upperLimit,lowerLimit,

SplitSim.thresholds.get(level1).getRandomState());

       if (s.run()){

SplitSim.thresholds.get(level).addHitCount();

SplitSim.thresholds.get(level).addState(s.getState());

}

else{

SplitSim.thresholds.get(level).addMissCount();

}

   }

if (level==thresholdCount) {

SplitSim.endSimulation();

   }

}

==========================================
```

The above algorithm can also be graphically represented by a flow chart as shown in Figure 3.

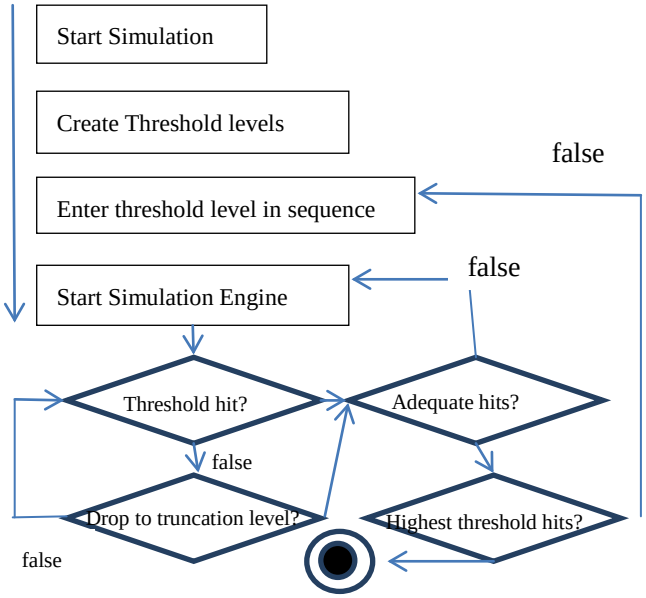

FIGURE 3 CORE ALGORITHM BEHIND SPLITSIM

Theoretical Foundation of the SPLITSIM Framework (Excess View of a System)

In mathematics, we have a concept of excessed numbers. An excess number is a number that its zero reference point is shifted to the right or to the left with the excess amount of value. The SPLITSIM framework uses the similar concept to shift our simulation focus.

The run time of most simulation software is spent evenly across the whole spectrum. For example, suppose that we select to study the likelihood of the nuclear power plant meltdown and to study the sequence of events that led to the rare event. The traditional simulation approach will be to design a general purpose simulation package and to run a series of simulations. To get some meaningful result, we must conduct the simulation for a really long period since this is a rare event. The concept of threshold splitting [2] suggests the speed of such simulation processes can be improved considerably. In a traditional simulation approach, the system is started at the optimal temperature and continues to run until the temperature crosses the maximum temperature level. The reason why the simulation takes so long in the traditional simulation approach is because a lot of simulation time is spent in the normal operational range of the system (i.e. the temperature range that is closer to $T_{optimal}$). For instance, if the reactor is designed to operate in its optimal temperature for 99.9% of the time, it translates to the same ratio of time that the simulation spends near the optimal threshold level. If the goal of the simulation is to study the behavior of the system in its normal operational mode, the traditional simulation approach works fine. However, that is not the case for the rare event systems.

The goal of a rare event simulation and the goal of the SPLITSIM framework are to simulate the behavior of the system in such a way that the rare events can be well studied. Thus, our framework needs to be designed in such a way that most of the simulation time will be spent near the proximity of the maximum threshold. That is exactly how our SplitSim Framework is designed. In other words, the theoretical foundation of the SPLITSIM can be summarized as the **skewed and magnified view** of a system. Specific splitting method that the SPLITSIM uses is the fixed number of success threshold level splitting [1]. The fixed number of success threshold level splitting is just a method to realize this theoretical foundation.

The Application of the Slitsim Tool to an Example

In this section, we show how the SplitSim framework works by applying it to an example. The example that we will use is a nuclear reactor system. We will give the problem specifications in the following section.

The Description of a Rare Event System Example

In a nuclear reactor system, the temperature of the core needs to be maintained at a certain temperature. So while there are factors in the reactor that increases the temperature, there are also wide ranges of sophisticated cooling mechanisms that cool the reactor. This results in a reactor system that for the most part maintains an optimum operating temperature of $T_{optimal}$. Since any temperature higher than $T_{optimal}$ moves the system to a less desired state, it can be considered a threshold (a boundary level). Hence the optimal temperature level can be also referred to as an optimal threshold of the system (i.e. a boundary level that defines the level of optimal or desired state).

Now let us consider a temperature of $T_{maximum}$, the maximum temperature at which a reactor meltdown occurs. This temperature level can be also referred to as the maximum threshold for the system (i.e. a boundary level that defines the maximum level supported by the system). To ensure that the meltdown does not occur, the system must be

designed in such a way that any rise of temperature is brought down eventually to its optimum operating temperature. In other words, the temperature of the core needs to be closest to the optimal threshold for the best case and lower than the maximum threshold for the worst case.

There will be times when unexpected things might happen during the operation of the reactor. For instance, a surge in the catalyst might trigger a sharp temperature rise. In such an event, the cooling mechanism works extra hard to cool the temperature. Let us assume that the cooling mechanism is pretty reliable and it has a very low probability of failure. So in almost all cases where the reactor runs into such a scenario, the cooling mechanism balances the temperature increase. However, unless the cooling mechanism is 100% reliable, there might exist an event when it fails right at the moment of a temperature surge. Furthermore, let us consider that there is a back-up cooling mechanism that starts once the original cooling mechanism fails. The back-up cooling mechanism also comes with a low probability of failure. Consider an event where there is a surge in temperature that results in gradual failure of both the original and the back-up cooling mechanism. Such an event could cause the temperature to exceed $T_{maximum}$, hence causing the rare event to occur.

The Application of the SplitSim Framework to the Example

In this section, we introduce aspects of applying our SPLITSIM software to the nuclear reactor example. Through the process, we will show features, and the graphic user interface of the SPLITSIM software package. In the following sections, we will describe the parameters of the sample runs and the interpretation of the results.

The Sample Runs of the SplitSim Simulation

In this section, we will show the result of some sample runs of applying our SplitSim framework to the nuclear reactor system. SplitSim GUI is pretty intuitive that makes its operation very straightforward and easy to understand. SplitSim UI consists of four sub panels: Start, Graph Data, Result Data and Help. To start the simulation one must enter all simulation parameters (Number of Splits, Truncation Bias, Number of hits per threshold level). The user can then start the simulation by pressing on 'Start Simulation Button'. This starts the simulation and the real-time display of the threshold incline starts in the Start Panel. Upon completion of the simulation, the probability of the rare-event is displayed in the Start Panel. Figure 4 shows the Start Screen of the SplitSim once the simulation has been started.

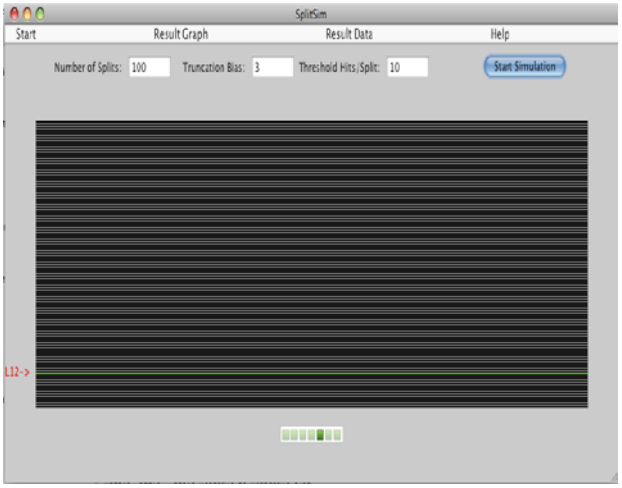

FIGURE 4 SNAPSHOT OF SPLITSIM START SCREEN

The black rectangular section in the screen represents a vertical simulation space with the grey horizontal lines representing the thresholds within the simulation space. The snapshot of SplitSim was captured when simulation was at its 12th threshold level. The green horizontal line with the red text on the left denotes the current threshold level in the simulation.

Once the simulation is complete, progress bar disappears and the window displays the probability of the rare event. The user can then click on the Result Graph Panel to view the graph of probability distribution over the threshold levels. The Result Graph Panel shows how likely is the system to reach the next threshold level from the current threshold. Figure 5 shows the probability distribution of the Reactor1 from level 0 to level 100.

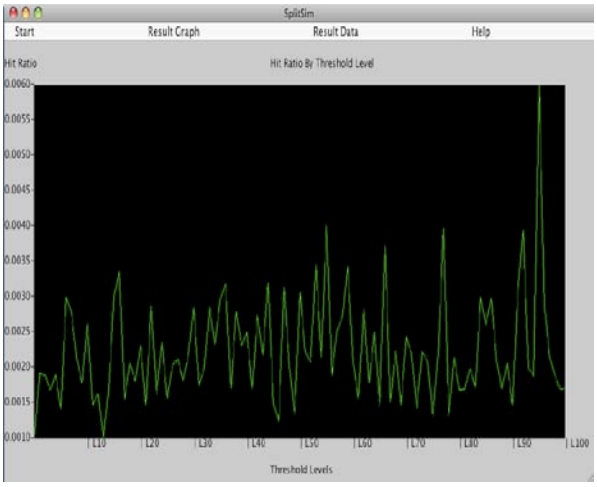

FIGURE 5 SNAPSHOT OF SPLITSIM RESULT GRAPH SCREEN

The next step is to click on the Result Data Panel. The Result Data Panel displays the probability distribution data in detailed tabular format. This data is particularly useful when comparing the level of risk tolerance at each threshold level. Figure 6 shows the Result Data screen for simulation of Reactor1. It shows the hit count, miss count, probability of each level and the overall probability at all levels within the simulation.

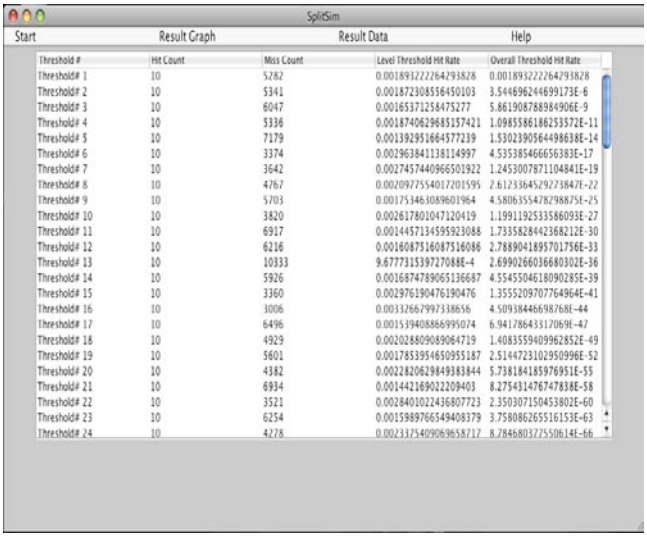

FIGURE 6 SNAPSHOT OF SPLITSIM RESULT DATA SCREEN

The Level of Maximum Interest screen displays the details of the system state for the threshold level that has highest level on contribution to the rare event occurrence. This is particularly useful when the risk tolerance of the system is to be improved. For instance when the nuclear reactor for Reactor1 was simulated, it was observed that the 74th level had the highest contribution towards the rare event occurrence.

In this case we conduct the simulation using the following test case:

Number of thresholds: 100

Truncation Level: 3

Number of successes per threshold: 10

Results and Findings

SplitSim framework comes packaged with three basic sample systems namely Reactor1, Reactor2 and Reactor3. The three systems represent a basic system to mock a nuclear reactor system that has attempts to keep its core temperature close to 300°F. The rare event for all three designs will be an event when a temperature of 400°F is reached. After running our newly developed software to the sample systems, we have the following findings and insights.

Truncation Bias Findings

The SplitSim framework offers truncation of downward bound runs. It eliminates the time spent on runs that have veered away from the threshold level enough to consider it a failure. This enables SplitSim to focus more time on runs that are more likely to contribution towards the rare event occurrence. However as mentioned in previous sections, it introduces a little bias in the results.

To better understand this bias, a series of simulation runs are performed in the example system of Reactor1. The simulation was started from a truncation level of 100. This corresponds to a truncation ration of 100%. The simulation was then continued with truncation levels 10% apart (i.e. 90,80,70….10). The time for the rare event simulation is recorded as well as the rare event probability that is calculated by SplitSim.

A graph is then plotted to compare the bias introduced with the level of truncation. The Truncation Bias for Rare Event is derived by the following formula:

Truncation Bias at Truncation level x = (Estimator at level x - Estimator at 100% truncation) / Estimator at 100% truncation

And Truncation Ratio is derived by:

Truncation Ratio at Truncation level x = x / Total number of threshold levels

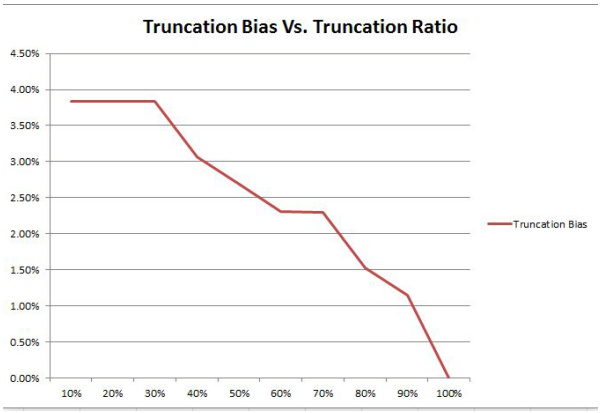

FIGURE 7 TRUNCATION BIAS VS. TRUNCATION RATIO

As predicted, the graph shows decrease of bias as the truncation level increases to 100% (i.e. no truncation). However the level of bias it brings is considerably low when compared to the magnitude of probability of the

rare event. A detailed graph of the findings can be seen above in Figure 7.

Levels of Interest Analysis

In this paper we define a new term, levels of interest in threshold, as the threshold levels that have higher contribution to the rare event occurrence when compared to the other levels. The levels of interest can be really useful in determining weak points in the system. SplitSim helps determine such levels of interest in a graphical method. The data can then be used to make improvements targeting just that portion of operation.

Knowing the Levels of Interest provides potential of savings in terms of resources and time required to improve the fault tolerance of the system. SplitSim comes with three reactor design systems: Reactor1, Reactor2 and Reactor3. The three systems have been purposed designed to behave differently at different stages within its operation.

The test cases will include the three sample nuclear reactor systems. Reactor3 has been designed to mock a reactor system that has uniformly acting cooling system. On the other hand Reactor1 and Reactor2 have been designed to mock a reactor system with a less efficient cooling system at higher temperatures and lower temperatures respectively.

The problem statement in this analysis section is to find out if SplitSim detects such behavior difference in the three alternative designs. So to solve for this problem, the three reactors were simulated one after another and its Result Graph screen was captured for analysis. The screen captures are listed in Figure 8, 9 and 10 for further analysis.

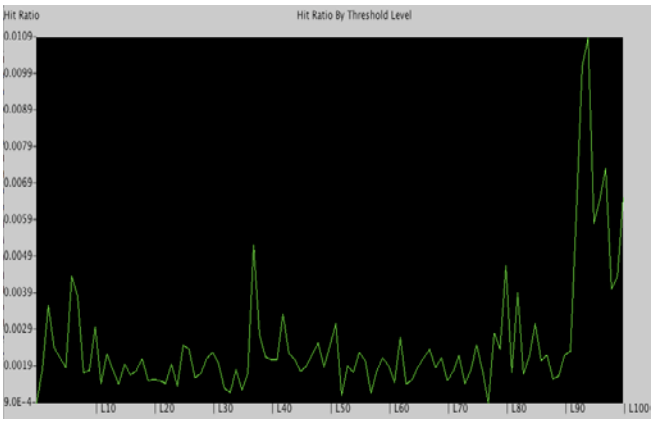

FIGURE 8 REACTOR 1 THRESHOLD RISE PROBABILITY GRAPH

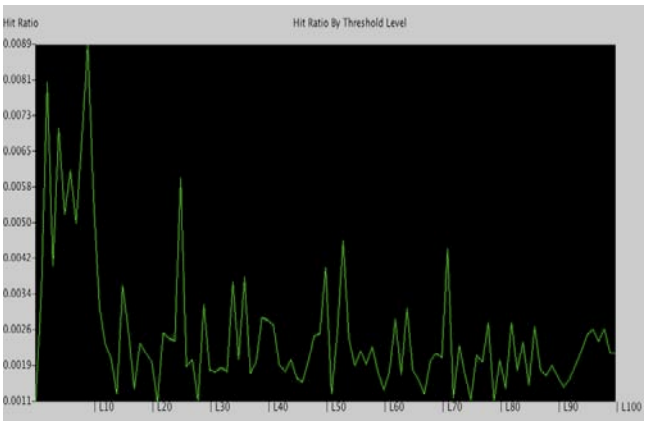

FIGURE 9 REACTOR 2 THRESHOLD RISE PROBABILITY GRAPH

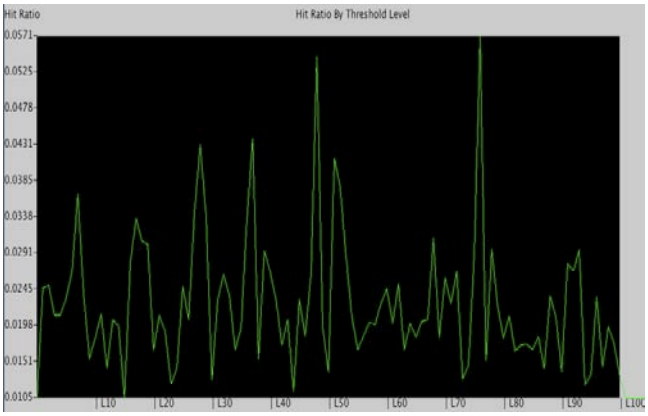

FIGURE 10 REACTOR 3 THRESHOLD RISE PROBABILITY GRAPH

It can be seen from comparing the Result Graph for the alternative reactor designs that they all show different pattern of threshold rise across the different threshold levels. The threshold rise probability graph for Reactor1 shown in Figure 8 reveals that the rare event prevention for Reactor1 is very poor in higher thresholds. The threshold rise probability graph for Reactor2 is very poor in the mid level thresholds as shown in Figure 9 as opposed to Reactor1 that has most of its levels of interest in the higher thresholds. Figure 10 on the other hand reveals somewhat uniform probably graph for Reactor3.

ACKNOWLEDGEMENTS

The second author likes to give thanks to his family member for the support for this writing project. Give thanks to Enlu Peng, Yuqing Peng, and Daniel Jian.

Note: this paper is based on the research work related to the first author's thesis.

REFERENCES

[1] Michael Amrein and Hans R. Künsch. 2011. A variant of importance splitting for rare event estimation: Fixed

number of successes. ACM Trans. Model. Comput. Simul. 21, 2, Article 13 (February 2011), 20 pages.

[2] Pierre L'Ecuyer, Val Demers, and Bruno Tuffin. 2006. Splitting for rare-event simulation. In Proceedings of the 38th conference on Winter simulation (WSC '06), L. Felipe Perrone, Barry G. Lawson, Jason Liu, and Frederick P. Wieland (Eds.). Winter Simulation Conference 137-148.

[3] Paul Glasserman, Philip Heidelberger, PerwezShahabuddin, and Tim Zajic. 1996. Splitting for rare event simulation: analysis of simple cases. In Proceedings of the 28th conference on Winter simulation (WSC '96), John M. Charnes, Douglas J. Morrice, Daniel T. Brunner, and James J. Swain (Eds.). IEEE Computer Society, Washington, DC, USA, 302-308.

[4] Manuel Villen-Altamirano and Jose; Villen-Altamirano. 1994. Restart: a straightforward method for fast simulation of rare events. In Proceedings of the 26th conference on Winter simulation (WSC '94), Mani S. Manivannan and Jeffrey D. Tew (Eds.). Society for Computer Simulation International, San Diego, CA, USA, 282-289.

[5] PerwezShahabuddin. 1995. Rare event simulation in stochastic models. In Proceedings of the 27th conference on Winter simulation (WSC '95), Christos Alexopoulos and Keebom Kang (Eds.). IEEE Computer Society, Washington, DC, USA, 178-185.

[6] John F. Shortle and Chun-Hung Chen. 2008. A preliminary study of optimal splitting for rare-event simulation. In Proceedings of the 40th Conference on Winter Simulation (WSC '08), Scott Mason, Ray Hill, Lars M\&\#246;nch, and Oliver Rose (Eds.). Winter Simulation Conference 266-272.

[7] Werner Sandmann. 2005. Importance sampling in Markovian settings. In Proceedings of the 37th conference on Winter simulation (WSC'05). Winter Simulation Conference 499-508.

[8] Pierre L'ecuyer, Jose H. Blanchet, Bruno Tuffin, and Peter W. Glynn. 2010. Asymptotic robustness of estimators in rare-event simulation. ACM Trans. Model. Comput. Simul. 20, 1, Article 6 (February 2010), 41 pages.

Author Introduction

Sudeep Kumar Shrestha Sudeep holds a Master of Science degree in Computer Science from Metropolitan State University, U.S.A. His primary research interests are in the field of simulation, object oriented databases and mobile application development.

Kuodi Jian Kuodi holds a Ph.D in Computer Science and Operations Research at North Dakota State University, Fargo, North Dakota. Computer System Architect at Banner Health System, Fargo North Dakota. Assistant Professor at Metropolitan State University, Saint Paul, Minnesota. His research interests include real-time operating systems, Database and Data structures, Operations Researches, Intelligent computing systems, Bioinformatics, and Simulations.